Public Integrated Modular Avionics (PIMA)*(proposed)*

A safety critical distributed real-time infrastructure
by

Infinite Delta Corp ∞∆
https://www.infinitedelta.com/wp/pima.pdf

1 Introduction

Public Integrated Modular Avionics (PIMA)*(proposed)* is a safety critical distributed real-time infrastructure. It includes a Real-Time-Operating-System (RTOS) designed to support the highest DO178C Design Assurance Levels (DAL). It encompasses the best features found in IMA[A], Capability Maturity Model Integration (CMMI) [B], and in the public domain.

Safety critical infrastructure is priority for avionics, medical, automotive, industrial, and agriculture. The FAA is currently under pressure from the growing UAV industry to support an effective infrastructure. Medical is facing relatively new inter system liability rules and high altitude issues. Self guided (autopilot) is now standard in industrial and agriculture applications, and is being demonstrated for automotive.

Infinite Delta Corp is proposing a public domain safety critical infrastructure. Many industries have already turned to the public domain for stable and secure infrastructure.

Positive feedback from the Modular Tickless Prioritized Preemptive RTOS[C] and Architecture for Safety[D] papers has led to the development on the Xilinx Zynq-7000 SOC[E] platform and a Zynq UltraScale+ MPSoc[F] future.

Contents

2 Current Status

Pima kernel was demonstrated on the Xilinx Zynq-7000 in 2015, but recent DDB changes broke the ARM build. Any potential release is pending a client's review and acceptance of PIMA's architecture, certification procedures and Single Event Upsets (SEU aka Soft Errors^G) solution. Detailed status are as follows:

- Kernel written in C++ using GCC 5.1.0.
- Dual ARM processors are running independent process lists.
- Pima events calls application class methods (Ada package functions) for processing.
- Tickless prioritized preemptive kernel has 60 percent test coverage.
- Events (preemptive and non-preemptive) have 30 percent test coverage.
- Gnat's Ada is operational with the runtime library (RTL) disabled.
- Missing DO178C level B architecture pre-qualification artifacts.
- Distributed Database (DDB) is currently under startup integration.

3 Clarifications

Terms used in this document are not necessary universal, so they are clarified in the following subsections for the reader.

3.1 Abortable processes

A process that has been either aborted, or preempted by one of its own defined preemptive events, will not resume where aborted. Instead, the scheduler will restart the aborted event after the preempted processing has completed.

3.2 Copyless

Pima minimizes copying data around by passing record structures by index or address. Even the logging system doesn't do anything with the format string. This also implies there must be two or more history records, and there is a limited amount of time the record is expected to be valid. A key function of the architect is to size each DDB table for the system.

3.3 Debian Linux Development Environment

Pima and PIMA's application development is on a Debian Linux Development Environment^H. Debian has the best life cycle practices with solid security and a far more commercial friendly licensing and repackaging.

3.4 Device Drivers

PIMA's device drivers have little resemblance to traditional device drivers. Each device driver runs entirely in their own protected memory space, with very limited time allocated. They are given limited access to only the hardware they need and may require an interrupt event. There isn't global access or privileges beyond a standard application.

3.5 Modular

The term **Modular** is the IMA[A] modular definition, which is to allow multiple vendors on a single platform. The Tickless Prioritized Preemptive RTOS uses ordinary GCC built and linked executables for loading and running. The executables are not identified by a common or build name, but by their digital signature.

3.6 Network Centric

Pima is Ethernet network centric, including Time-Triggered Ethernet[I] (SAE AS6802[J]) for IMA fault-tolerant and all forms of testing and diagnostics. Any serial IO is strictly for any applications.

3.7 Object oriented paradigm

Pima fully supports the object oriented paradigm. Pragmatically this results in using class methods being called instead of setting up call back routines. Ada applications will need to supply catch routines for a pre-defined or pre-built package. Ada, C++, and DO178C supports this object oriented encapsulation, even under the most restricted environments.

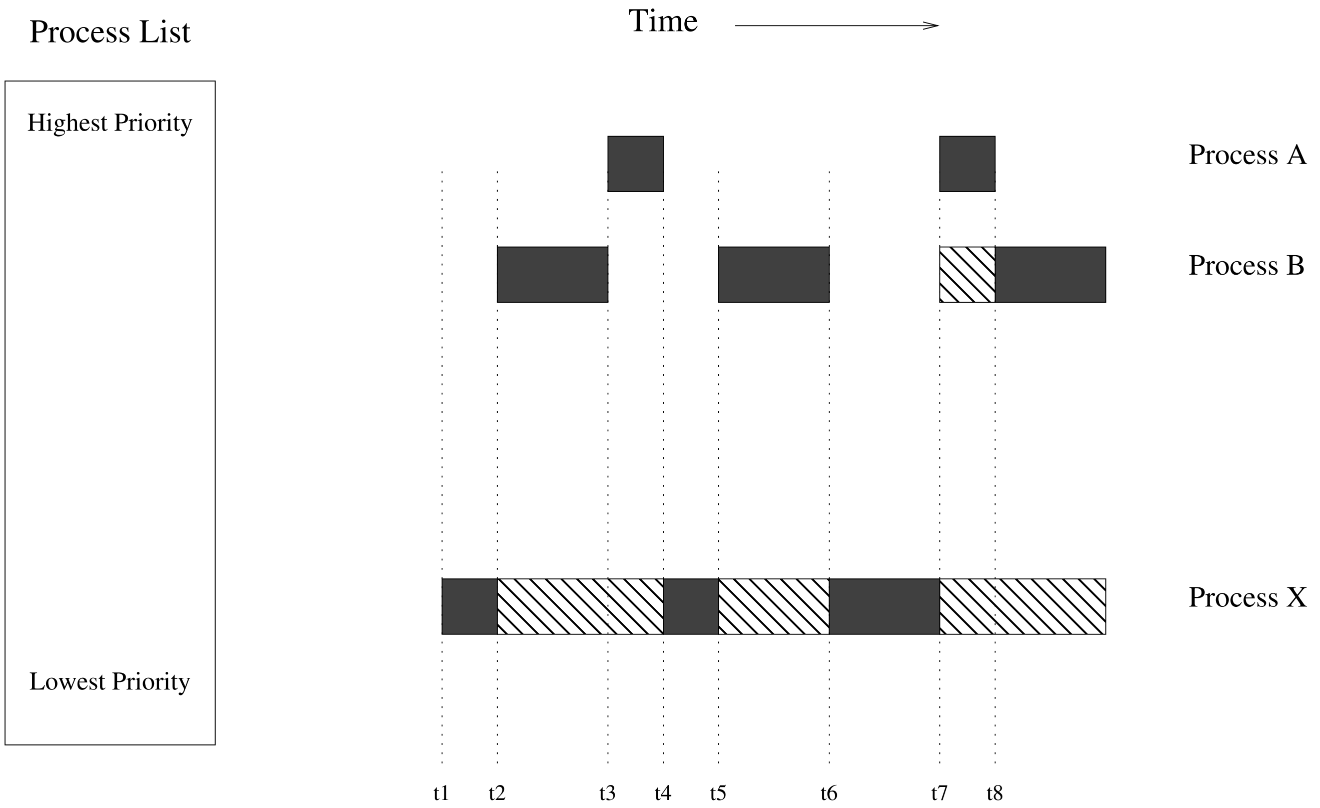
3.8 Restricted Environment

Every certification organization has limits to what features an application is allowed to use. This largely depends on the application's DAL safety level. Memory allocation, dynamic dispatching, and recursion are the first features to be eliminated. This eliminates most of Ada's Run Time Library (RTL) and almost all of C++ standard libraries and any use of virtual functions.

3.9 Tickless Prioritized Preemptive RTOS

The Tickless Prioritized Preemptive RTOS is a simple Prioritized Preemptive RTOS with the processes sorted by priority:

Tickless Prioritized Preemptive RTOS



Higher priority processes will always preempt lower priority processes. A **Tickless** system only will perform process scheduling when processes are done, or a higher process wants to run.

Example, at time *t1* the scheduler starts at the highest priority process and looks down until it finds a process to run. It found Process X to run, but before X is run, a hardware timer is set to wake when Process B wants to run. This gives processes a high resolution control over when they start and end. Runtime is illustrated as a dark bar, while wait times are shown as hatched bars.

Note how the higher priority applications interrupt lower priority applications. Process B interrupts Process X at *t2 and t5* then Process A interrupts Process X at *t3* and both Processes B and X at *t7*.

4 Architecture

Pima uses many separately loadable modules for any given project. Most modules will be reused across many projects such as the network module. A digitally signed load table determines which modules are loaded, their priorities, and restrictions. Each module is identified only by its digital signature.

Full use of the virtual memory controller (MMU) is used to keep tight control of all memory accesses to the system.

4.1 Digital Signatures

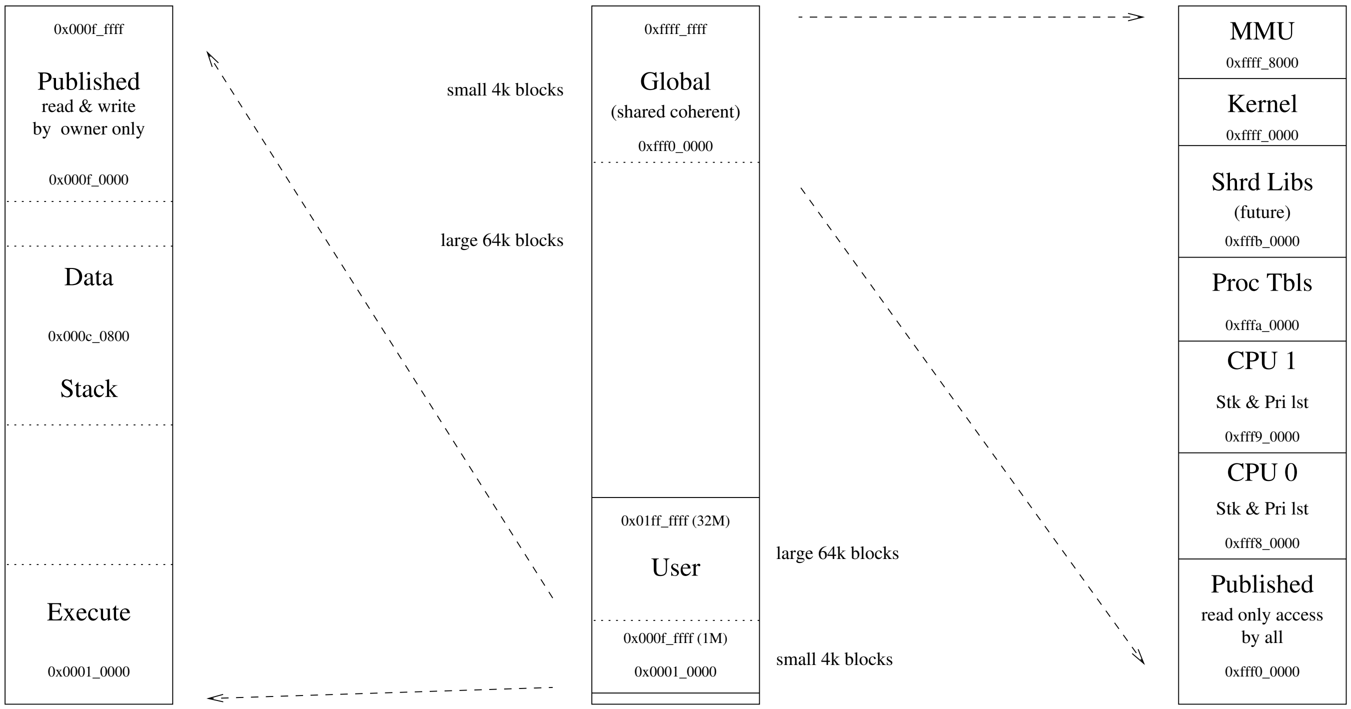
Digital signatures are used to identify objects in the system, including modules, data structures, and configurations. Restrictions include:

- Although a module is built under a common name, it is only identified by it's digital signature.
- The module digital signature may not include the entire build image, only the sections loadable into the Pima target.
- No version identifiers and/or build timestamps are allowed in the the loadable images.
- Certification life cycles are handled by the signature covering the load table.

4.2 Virtual Memory Map

The lower one (1) megabyte of memory is dedicated for user small 4k blocks and thirty two (32) megabytes for large user data. The upper one (1) megabyte of memory is global and for the kernel.

PIMA Virtual Memory Map



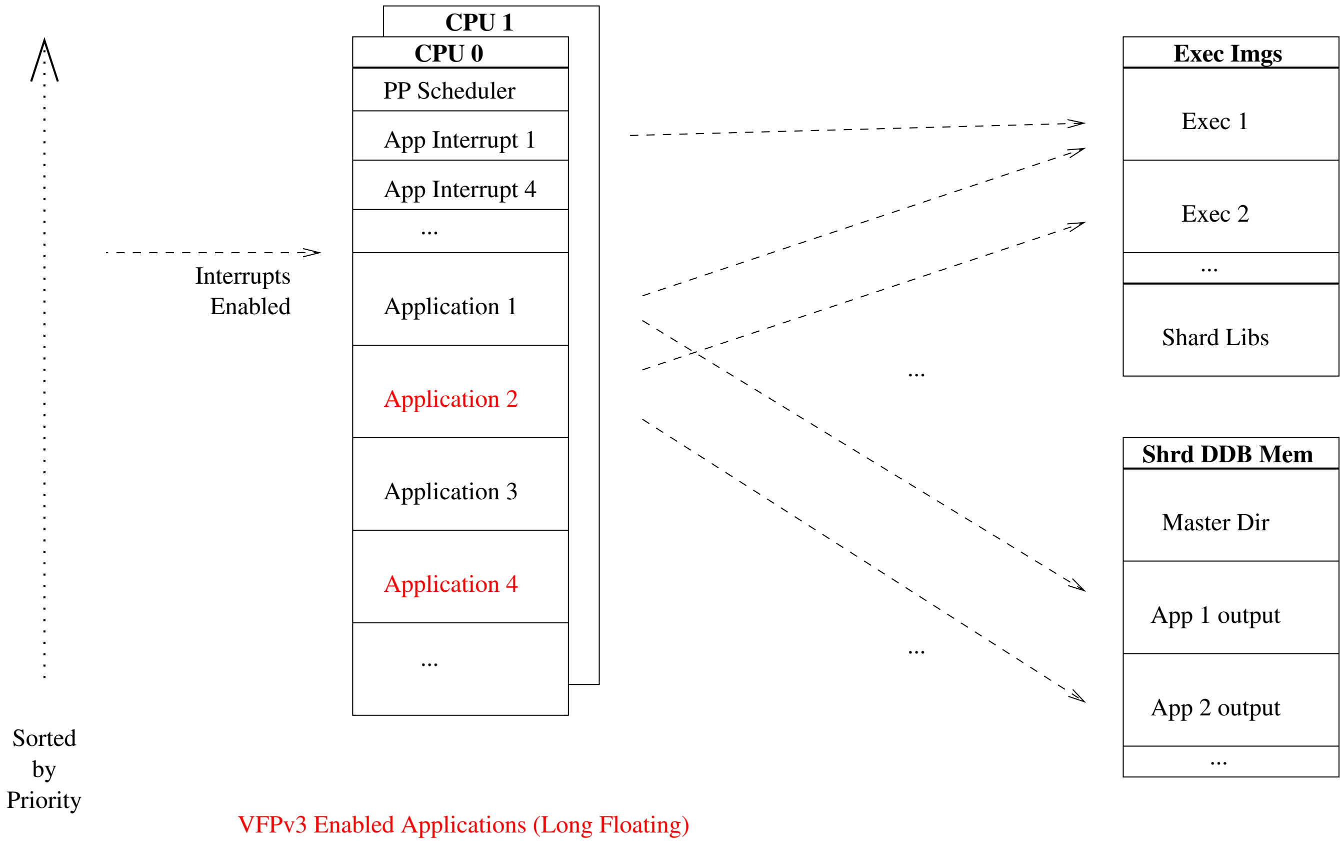
All inter-process communications are through published memory areas. A read-only copy of the user's published memory space is available for all users to read. This prevents any process from interfering with any other process without any need for kernel run time support.

A stack overflow is now detected by the hardware. There is no need to enable software stack protections.

4.3 Process Architecture

Processes are sorted by priority and are guarded from overrunning their allocated time. Interrupts are only high priority processes with their allocated CPU usage limits.

PIMA Process Architecture



Device drivers are only applications. Device drivers are subject to same individual time and space restrictions as any other application.

The architecture could support light and heavy threads, but there isn't any kernel support at this time. Events can be preemptive within a thread.

4.4 Event Architecture

Pima supports both preemptable and abortable processes. Processes can be aborted out of their current task without any attempt to resume. Instead the process is restarted and then allowed to complete the task from the beginning.

Each process can have the Pima scheduler monitor many events. Events can be configured as synchronous to avoid aborting or other threading issues. Events can also be configured as preemptive which can abort the current process thread.

4.4.1 Single Event Upsets (SEU) problem and solution

Early processors considered for safety applications only use parity on its cache. The issue is SEU's occur 300 times more often at an altitude of 40,000 feet. This could translate into several times a mission depending on the processor.

Solutions include disabling cache, add OS/application support, or use processors which support ECC on cache. Freescale's MPC7448, Altera SOC, and Zynq® UltraScale^{tm}+MPSOCs (ARM Cortext-A53) all have ECC even on its cache.

4.4.2 Event Mechanism

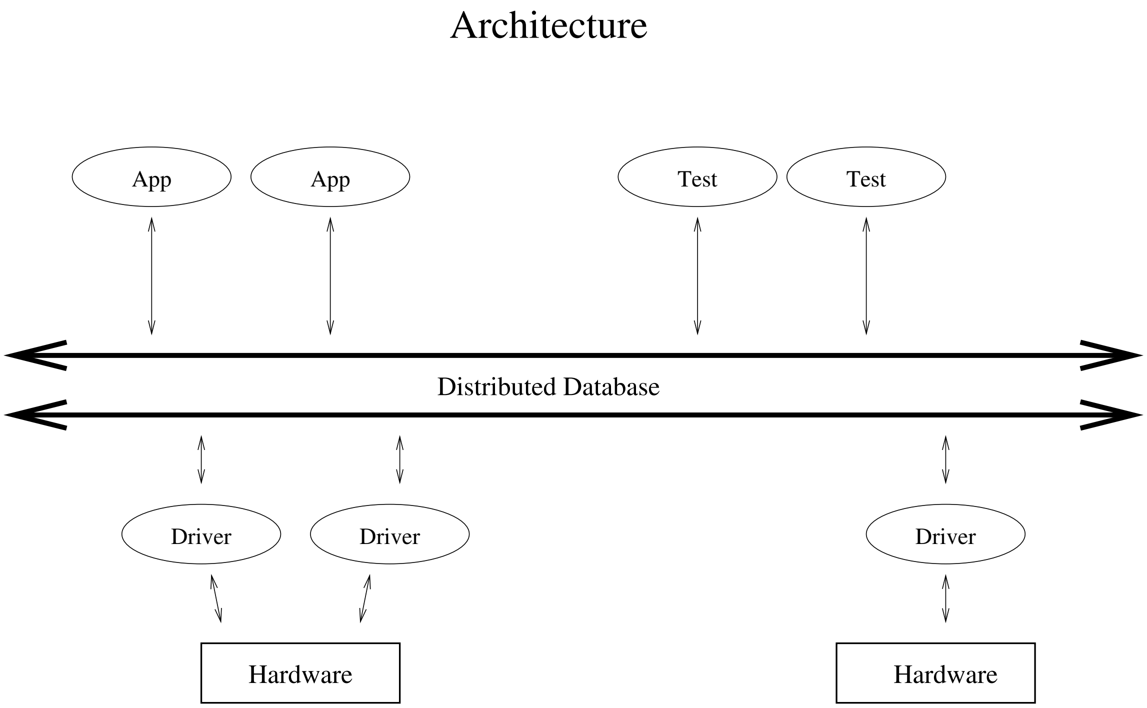
Monitoring a single 32 bit word for change is the core Event mechanism:

```
bool Active(){if ((EventSnap = *WatchAddress) != LastEvent) {++StartCount; return true;} re
```

It supports all interprocess communications, different starts (like cold starts, warm starts, etc), state changes, and some critical hardware events including interrupts.

4.5 Distributed Data Base (DDB) Architecture

A Distributed Data Base (DDB) architecture is used for all intra and inter CPU and networked systems. DDB is light weight, using a Copyless interface. DDB departs from most database applications in that it supports high performance deterministic applications.



DDB supports Test Driven Development (TDD) processes even when hardware is not available. DDB is also network centric for all its core activities. See Architecture for Safety^D for an operational example.

4.6 Scheduler

PIMA's scheduler is the heart of the kernel. The scheduler runs through a list of processes and checks to see if they are "Ready to Run". This function determines if a process needs to run, and for how long. If the process is not actually ready to run, and there isn't another process wanting to run sooner, it sets next timer interrupt time. It is the scheduler that enforces all process time restrictions to guarantee system level deterministic behavior.

4.7 Device Drivers

Device drivers communicate only with Distributed Data Base (DDB), never directly with any applications. It receives and publishes data using common engineering units with the DDB. When more than one data object is supported, the device driver is table driven as to exactly how it converts to and from the DDB. These conversion tables may include reasonableness tests, however reasonable and rates tests shall be the primary responsibility of the consuming application.

Strict minimal logic standards are placed on device drivers. They are also required to maintain data cohesion into the DDB.

4.8 Logging and Internationalization

A common logging system for the kernel and all the applications is used per the public areas described earlier. However, no actual ASCII text is to be placed into the data structure. The logging record contains only the data, and if a format string is supplied, then only the address of the format string to be included. Actual formatting will not be performed until it is used. This eliminates almost all of the runtime overhead associated with logging, and several certification issues. More importantly it supports and simplifies internationalization with zero impact on critical applications.

This model is strongly encouraged down to the application level. All critical business level applications use a familiar call with a format string followed by data. Again, no actual formatting is allowed within the critical application.

The user application, which is less critical, and will have language specific support, can easily translate the format address into the language specific format string for the end user. The user application can also be anywhere, including on a properly authorized networked device (Smart Phone, etc).

5 Qualifications and Certification

Getting qualification and certification pre-acceptance on key architectural items is required. Pima combines the best practices from avionics and the public domain. DO178C level B certification depends on the acceptance of the following:

5.1 Without a qualified compiler

Using and depending on public domain software has some concerns for commercial and safety use. Licensing has become more limited and is being enforced. There is no concept of ‘‘Qualified Compiler’’ in the public domain. All qualification testing is performed on the final image. The public domain supports dozens of target processors, complete with a huge set of varied options. Final executable image testing can easily coincide with DO178C level A structural analysis. Hence, only DO178C level A structural analysis shall be used. All source code based structural analysis is no longer viable.

Pima strongly encourages the Test Driven Development (TDD) approach, its even used on its scheduler.

5.2 Test Driven Development (TDD)

TDD allows qualification testing through to the final target system. When properly performed, can led to complete DO178B level A object level structural coverage. The best benefit is that **TDD very effectively automates the author(s) detailed review**. All changes are effectively and immediately reviewed by the previous author(s) during development. Typically before committing changes to the source code library. TDD totally removes any feeling of module fragility, commonly associated with larger projects.

5.3 Prioritized Preemptive Certification

Process preemption is the leading concern for deterministic system operation. Strict control over preemption regains qualitative control for these systems. The Pima kernel places strict limits on each process.

Every process is required to have a worst case tests. This include CPU, IO, interrupts, and memory. Each process shall be limited by the kernel to its worst case, thus allowing the impact on any lower priority processes.

Each project will run these tests, in conjunction with the other applications worst case tests for acceptance. The project architects can then determine the system performance and qualify the configuration. The impact of processes sharing resources becomes visible early in a projects life.

Note, high amount of IO, or interrupts can have a dramatic effect on system timing requirements. Even the low priority non-essential applications require some resource modeling, if only to be modeled as a group.

Pima will be verbose on all performance issues. Pima tools can be certified to document expected and worse case timing. Pima logging can report any violations and quickly help resolve any race conditions.

5.4 Abortable Processes

Although todays computer platforms are for more reliable, there are problems which only rerunning an application can resolve. Processes being aborted shall be expected and planned for in the Pima development environment. Pima attempts to minimize the pain, but keeps the concern visible to all.

Every application thread must be tested for proper recovery after an abort.

5.5 ARM cache parity testing

Xilinx Zynq-7000 ARM does not support any run time cache parity testing. The cache parity operation must be acceptable at a different level, such as at part certification. A certification Pima/Zynq-7000 parity test is required at Los Alamos Neutron Science Center (LANSCE) for full production release.

5.6 Distributed Data Base (DDB)

A Distributed Data Base (DDB) shall be used for all inter process, inter processor, and inter system communication and configuration.

5.7 Digital Signatures as identifiers

Digital Signatures or Hashes shall be used for all loading, configuration, and approvals. Accepting Digital Signatures as the unique identifier for each image or structure, allows both automated and rigorous regression testing.

5.8 Qualified Project Architect Tools

Pima requires a qualified project architect tool(s) to quantify a given system configuration determinism. These tool(s) at a minimum must add all potential process preemption to determine performance.

6 Conclusion

Public Integrated Modular Avionics (PIMA)*(proposed)* is a safety critical distributed real-time infrastructure. Although the kernel is stable, there are some core infrastructure required before initial releasing. The infrastructure supports project planning through integration, but can be especially valuable into integration, and product support.

There is a shortage of mission critical experienced engineers, Pima is ideally setup to support engineering educational institutions to help deliver these resources.

7 References

A. Integrated Modular Avionics (IMA)
`http://en.wikipedia.org/wiki/Integrated_modular_avionics`
`https://www.researchgate.net/publication/235731222_Model-based_Development_of_Integrated_Modular_Avionics_Architectures_on_Aircraft-level`

B. Capability Maturity Model Integration
`http://en.wikipedia.org/wiki/Capability_Maturity_Model_Integration`

C. Modular Tickless Prioritized Preemptive RTOS
`https://www.infinitedelta.com/wp/avionics_rtos.pdf`

D. Architecture for Safety
`https://www.infinitedelta.com/wp/architecture_for_safety.pdf`

E. Xilinx Zynq-7000
`http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html`

F. Xilinx UltraScale+ MPSoC
`https://www.xilinx.com/products/boards-and-kits/zcu106.html`

G. Single Event Upsets (Soft Errors)
`http://en.wikipedia.org/wiki/Soft_error`
`http://www.xilinx.com/support/documentation/user_guides/ug116.pdf`

H. Debian Linux
`https://www.debian.org/intro/free`

I. Time-Triggered Ethernet (SAE AS6802) (also known as TTEthernet or TTE)
`https://en.wikipedia.org/wiki/TTEthernet`

J. SAE International SAE AS6802
`https://www.sae.org/standards/content/as6802/`